Sylvester 方程 *AXB+CXD=E* 的共轭梯度法

董 宁，余 波

（湖南工业大学 理学院，湖南 株洲 412008）

摘 要：对 Sylvester 方程 ***AXB+CXD=E*** 提出了一种共轭梯度算法及 2 种预处理算法，讨论了算法的性质。数值试验表明，共轭梯度法适合解决大规模问题，预处理方法能有效地减少迭代次数。

关键词：Sylvester 方程；共轭梯度；预处理

中图分类号：O29 **文献标志码**：A **文章编号**：1673-9833(2010)02-0018-04

A Conjugate Gradient Method for Sylvester-Type Equation *AXB+CXD=E*

Dong Ning，Yu Bo

（School of Science，Hunan University of Technology，Zhuzhou Hunan 412008，China）

Abstract：Proposes a conjugate gradient method and two preconditioned methods for solving Sylvester-type equation ***AXB+CXD=E*** and discusses the properties of the algorithms. The numerical experiments show that the proposed conjugate gradient method is efficient to solve large scale problems and the preconditioned methods can reduce the number of iterations effectively.

Keywords：conjugate gradient method；Sylvester-type equation；preconditioned method

0 引言

考虑 Sylvester 型方程：

$$AXB+CXD=E, \tag{1}$$

其中：A，B，C，D 和 E 是 $n\times n$ 实矩阵；X 是待求的 $n\times n$ 实矩阵。方程（1）解的存在性研究见文献[1]。

最近，文献[2]对方程 $AXB=C$ 提出了共轭梯度（CG）形式的算法，这种算法保证了在不同的迭代步中方程的残差是正交的，从而能在有限的迭代中获得方程的最小二乘解。文献[3]进一步讨论了文献[2]中方法的收敛率与矩阵 A 和 B 的最大（小）奇异值有关。对另一 Sylvester 型方程 $AXB+CX^{\mathrm{T}}D=E$，文献[4]提出了最速下降型的算法并讨论了算法的收敛性质。笔者对方程（1）提出一种共轭梯度形算法。类似于文献[2]，在不同的迭代步中，由这种算法获得的残差相互正交，因此算法是适定的（well-defined），并且适合于解决大规模的问题。正如通常的共轭梯度形式算法一样，这种算法通常需要较多的迭代次数，因此我们给出了 2 种预处理算法来有效地减少原算法的迭代次数。Matlab 中数值试验验证了上述的结论。

在本文中，记 I_n 为 $n\times n$ 的单位矩阵，记 $\|A\|_{\mathrm{F}}$ 和 $\mathrm{tr}(A)$ 分别为 $n\times n$ 矩阵 A 的 Frobenious 范数和迹，记 $\mathrm{vec}(A)=(a_1^{\mathrm{T}},\cdots,a_n^{\mathrm{T}})^{\mathrm{T}}$，其中 a_i 为矩阵 A 的第 i 列。$A\otimes B$ 为矩阵 A 和 B 的 Kronecker 积。若 $tr(AB)=0$，则称矩阵 A 和 B 正交。对于 n 维实向量 c，$\|c\|$ 表示其欧氏范数。

1 算法及性质

考虑到方程（1）可能没有解，转而求方程的最小二乘解。即考虑如下问题：

收稿日期：2009-09-10

通信作者：董 宁（1979-），女，山东菏泽人，湖南工业大学教师，主要研究方向为应用数学，图像处理，
E-mail：dnsx216@tom.com；
余 波（1979-），男，湖南株洲人，湖南工业大学教师，湖南大学博士生，主要研究方向为计算数学.

问题1　给定矩阵 A，B，C，D 和 $E\in\mathbb{R}^{m\times n}$。记 $f(X)=\|E-AXB-CXD\|_F^2$。求最小二乘解 $X^*\in\mathbb{R}^{n\times n}$，使得

$$f(X^*)=\min_{X\in\mathbb{R}^{n\times n}} f(X)。$$

利用矩阵迹的性质 $\mathrm{tr}[AB]=\mathrm{vec}(A^{\mathrm{T}})^{\mathrm{T}}\mathrm{vec}(B)$，求解方程（1）等价于找最小二乘解 $x^*\in\mathbb{R}^{n^2\times 1}$，使得

$$\|e-Gx^*\|_2=\min_{x\in\mathbb{R}^{n^2\times 1}}\|e-Gx\|, \tag{2}$$

其中：$G=B^{\mathrm{T}}\otimes A+D^{\mathrm{T}}\otimes C$，$x=\mathrm{vec}(X)$，$e=\mathrm{vec}(E)$。问题（2）的正规方程（normal equation）为

$$G^{\mathrm{T}}Gx=G^{\mathrm{T}}e, \tag{3}$$

由最小二乘理论[5]知，正规方程（3）能改写成如下矩阵方程形式：

$$A^{\mathrm{T}}QB^{\mathrm{T}}+C^{\mathrm{T}}QD^{\mathrm{T}}=0, \tag{4}$$

其中 $Q=E-AXB-CXD$。基于上述方程（4），给出如下共轭梯度算法来解方程（1）。

算法1　（CG算法）

Step0　给定初始矩阵 $X_0\in R^{n\times n}$，记

$Q_0=E-AX_0B-CX_0D$，

$R_0=A^{\mathrm{T}}Q_0B^{\mathrm{T}}+C^{\mathrm{T}}Q_0D^{\mathrm{T}}$，$P_0=R_0$，$k:=0$。

Step1　如果 $R_k\neq 0$，则

$$\alpha_k=\frac{\|R_k\|_F^2}{\mathrm{tr}\left[(AR_kB+CR_kD)^{\mathrm{T}}AP_kB+CP_kD\right]},$$

$X_{k+1}=X_k+a_kP_k$，

$Q_{k+1}=E-AX_{k+1}B-CX_{k+1}D=$

$Q_k-\alpha_k(AP_kB-CP_kD)$，

$$R_{k+1}=A^{\mathrm{T}}Q_{k+1}B^{\mathrm{T}}+C^{\mathrm{T}}Q_{k+1}D^{\mathrm{T}}=$$
$$R_k-\alpha_k\left[A^{\mathrm{T}}(AP_kB+CP_kD)B^{\mathrm{T}}+C^{\mathrm{T}}(AP_kB+CP_kD)D^{\mathrm{T}}\right],$$

$$\beta_{k-1}=\frac{\mathrm{tr}\left[(AP_kB+CP_kD)^{\mathrm{T}}(AR_{k+1}B+CR_{k+1}D)\right]}{\|AP_kB+CP_kD\|_F^2},$$

$P_{k+1}=R_{k+1}+\beta_{k-1}P_k$，

$k:=k+1$。

下面的引理给出了算法1中 R_i，P_i 的一些性质。

引理1　设矩阵 R_i，P_i（$i,j=0,1,2,\cdots,k$）由算法1产生，则对于所有的 $i\neq j$ 有

$$\mathrm{tr}\left[R_i^{\mathrm{T}}R_j\right]=0, \tag{5}$$

$$\mathrm{tr}\left[(AP_iB+CP_iD)^{\mathrm{T}}(AP_jB+CP_jD)\right]=0。 \tag{6}$$

证明　用归纳法证明。对于 $j=1$，由 α_0，β_1，R_0 的定义有

$$\mathrm{tr}\left[R_0^{\mathrm{T}}R_1\right]=\|R_0\|_F^2=$$
$$\alpha_0\mathrm{tr}\left[(AR_0B+CR_0D)^{\mathrm{T}}(AP_0B+CP_0D)\right]=0,$$
$$\mathrm{tr}\left[(AP_0B+CP_0D)^{\mathrm{T}}(AP_1B+CP_1D)\right]=$$
$$\mathrm{tr}\left[(AP_0B+CP_0D)^{\mathrm{T}}(AR_0B+CR_0D)\right]+$$
$$\|AP_0B+CP_0D\|_F^2=0。$$

因此式（5）和式（6）对 $j=1$ 成立。

现假设对所有 $j=s\le k-1$ 和 $i<s$ 有 $\mathrm{tr}\left[R_i^{\mathrm{T}}R_s\right]=0$ 和 $\mathrm{tr}\left[(AP_iB+CP_iD)^{\mathrm{T}}(AP_sB+CP_sD)\right]=0$ 成立，将证明式（5）和式（6）对所有 $i\le k$，$j\le k$，$i\neq j$ 成立。由算法1的 **Step0** 有

$$\left[R_i^{\mathrm{T}}R_{s-1}\right]=\mathrm{tr}\left[R_i^{\mathrm{T}}R_s\right]-$$
$$\alpha_s\mathrm{tr}\left[(AR_iB+CR_iD)^{\mathrm{T}}(AP_sB+CP_sD)\right]-$$
$$-\alpha_s\mathrm{tr}\left[(AP_iB+CP_iD)^{\mathrm{T}}(AP_sB+CP_sD)\right]+$$
$$\alpha_s\beta_i\mathrm{tr}\left[(AP_{i-1}B+CP_{i-1}D)^{\mathrm{T}}(AP_sB+CP_sD)\right]=0,$$

并且

$$\mathrm{tr}\left[R_s^{\mathrm{T}}R_{s+1}\right]=\|R_s\|_F^2=$$
$$\alpha_s\mathrm{tr}\left[(AR_sB+CR_sD)^{\mathrm{T}}(AP_sB+CP_sD)\right]=0,$$

再由算法1 **Step1** 中的 R_{i+1} 的定义有

$$\mathrm{tr}\left[(AP_iB+CP_iD)^{\mathrm{T}}(AP_{i+1}B+CP_{i+1}D)\right]=$$
$$\mathrm{tr}\left[(AP_iB+CP_iD)^{\mathrm{T}}(AR_{s+1}B+CR_{s-1}D)\right]=$$
$$\frac{1}{\alpha_i}\mathrm{tr}\left[(R_i-R_{i+1})^{\mathrm{T}}R_{s+1}\right]=0,$$

最后按 $j=1$ 时类似的方法可以得到

$$\mathrm{tr}\left[(AP_iB+CP_iD)^{\mathrm{T}}(AP_{s+1}B+CP_{s-1}D)\right]=0。$$

证毕。

引理1的式（5）说明算法1中的 α_k 和 β_k 的选取保证了方程（4）残差的正交性，式（6）则说明算法1产生的方向矩阵是相互共轭的。因此算法1将最多迭代 n^2+1 步终止到式（4）的解，从而是适定的。同时由式（5）可以得到 α_k 和 β_{k+1} 的较简洁的表达式：

$$\alpha_k=\frac{\|R_k\|_F^2}{\|AP_kB+CP_kD\|_F^2}, \tag{7}$$

$$\beta_{k-1}=\frac{\|R_k\|_F^2}{\|R_{k-1}\|_F^2}。 \tag{8}$$

事实上，由式（5）和 α_k 的定义可以直接得到式（7）。另外由 R_{k+1} 的定义有：

$$A^{\mathrm{T}}(AP_kB+CP_kD)B^{\mathrm{T}}+C^{\mathrm{T}}(AP_kB+CP_kD)D^{\mathrm{T}}=$$
$$\frac{1}{\alpha_k}(R_k-R_{k-1}),$$

因此 β_{k+1} 的分母可以改写成

$$\mathrm{vec}(\boldsymbol{R}_{k+1})^{\mathrm{T}}\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}\,\mathrm{vec}(\boldsymbol{P}_k)=\frac{1}{\alpha_k}\mathrm{vec}(\boldsymbol{R}_{k+1})^{\mathrm{T}}(\mathrm{vec}(\boldsymbol{R}_k)-\mathrm{vec}(\boldsymbol{R}_{k+1})),$$

于是由式（5）和式（7）可以得到式（8）。

下面的定理给出了算法1的收敛性，由于它和向量形式的共轭梯度算法收敛性一致（见文献[6]），故略去证明。

定理1 设$\boldsymbol{X}^*$是方程（4）的解。则由算法1产生的矩阵序列$\{\boldsymbol{X}_k\}$满足

$$\left\|\mathrm{vec}(\boldsymbol{X}_k)-\mathrm{vec}(\boldsymbol{X}^*)\right\|_{\boldsymbol{M}}\le 2\left(\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}\right)^k\left\|\mathrm{vec}(\boldsymbol{X}_0)-\mathrm{vec}(\boldsymbol{X}^*)\right\|_{\boldsymbol{M}},$$

其中$\boldsymbol{M}=\boldsymbol{G}^{\mathrm{T}}\boldsymbol{G}$，$\|\boldsymbol{x}\|_{\boldsymbol{M}}=(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{x})^{1/2}$，$\kappa$为$\boldsymbol{M}$的条件数。

我们可以进一步讨论条件数κ和系数矩阵之间的关系。

推论1 若$\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$和$\boldsymbol{D}$是对称半正定，则

$$\kappa\le\left(\frac{\sigma_{1A}\sigma_{1B}+\sigma_{1C}\sigma_{1D}}{\sigma_{nA}\sigma_{nB}+\sigma_{nC}\sigma_{nD}}\right)^2,$$

其中σ_{1A}（σ_{nA}）为矩阵$\boldsymbol{A}$的最大（小）奇异值。

证明 因为$\boldsymbol{A}$，$\boldsymbol{B}$，$\boldsymbol{C}$，$\boldsymbol{D}$对称半正定，由Weyl定理[7]，不等式$|\lambda(\boldsymbol{AB})|\le\sigma_{1A}\sigma_{1B}$和$\lambda(\boldsymbol{A}\otimes\boldsymbol{B})=\lambda(\boldsymbol{A})\lambda(\boldsymbol{B})$得

$$\lambda_1(\boldsymbol{M})\le\lambda_1(\boldsymbol{AA})\lambda_1(\boldsymbol{BB})+\lambda_1(\boldsymbol{CC})\lambda_1(\boldsymbol{DD})+2\lambda_1(\boldsymbol{BD})\lambda_1(\boldsymbol{AC})=(\sigma_{1A}\sigma_{1B}+\sigma_{1C}\sigma_{1D})^2,$$

其中λ和λ_1分别表示矩阵的特征值和最大特征值。同理利用不等式$|\lambda(\boldsymbol{AB})|\ge\sigma_{nA}\sigma_{nB}$[7]和Weyl定理，有

$$\lambda_n(\boldsymbol{M})\ge(\lambda_n(\boldsymbol{A})\lambda_n(\boldsymbol{B})+\lambda_n(\boldsymbol{C})\lambda_n(\boldsymbol{D}))^2-(\sigma_{1A}\sigma_{1B}+\sigma_{1C}\sigma_{1D})^2$$

其中λ_n表示矩阵的最小特征值。证毕。

3 预处理算法

上一节的算法1能有效地解决大规模的Sylvester方程，但是通常迭代次数较多。从定理2中可以看出有效地降低矩阵$\boldsymbol{M}$的条件数可以提高算法的收敛速度。类似于文献[6]，下面给出2种预处理CG算法来降低$\boldsymbol{M}$的条件数。

算法2（预处理CG算法）

Step0 给定初始矩阵$\boldsymbol{X}_0\in\mathbf{R}^{n\times n}$，设$\boldsymbol{Q}_0$和$\boldsymbol{R}_0$同算法1，选择1个对称正定的预处理矩阵$\boldsymbol{K}\in\mathbf{R}^{n^2\times n^2}$，记$\boldsymbol{Z}_0=\mathrm{vec}^{-1}(\boldsymbol{K}^{-1}\mathrm{vec}(\boldsymbol{R}_0))$，$\boldsymbol{P}_0=\boldsymbol{Z}_0$，$k:=0$。

Step1 如果$\boldsymbol{R}_k\ne 0$，则

$$\alpha_k=\frac{\mathrm{tr}[\boldsymbol{Z}_k^{\mathrm{T}}\boldsymbol{R}_k]}{\|\boldsymbol{AP}_k\boldsymbol{B}+\boldsymbol{CP}_k\boldsymbol{D}\|_F^2},$$

$$\boldsymbol{X}_{k+1}=\boldsymbol{X}_k+\alpha_k\boldsymbol{P}_k,$$

$$\boldsymbol{Q}_{k+1}=\boldsymbol{E}-\boldsymbol{AX}_{k+1}\boldsymbol{B}-\boldsymbol{CX}_{k+1}\boldsymbol{D}=\boldsymbol{Q}_k-\alpha_k(\boldsymbol{AP}_k\boldsymbol{B}+\boldsymbol{CP}_k\boldsymbol{D}),$$

$$\boldsymbol{R}_{k+1}=\boldsymbol{R}_k-\alpha_k[\boldsymbol{A}^{\mathrm{T}}(\boldsymbol{AP}_k\boldsymbol{B}+\boldsymbol{CP}_k\boldsymbol{D})\boldsymbol{B}^{\mathrm{T}}+\boldsymbol{C}^{\mathrm{T}}(\boldsymbol{AP}_k\boldsymbol{B}+\boldsymbol{CP}_k\boldsymbol{D})\boldsymbol{D}^{\mathrm{T}}]$$

$$\boldsymbol{Z}_{k+1}=\mathrm{vec}^{-1}(\boldsymbol{K}^{-1}\mathrm{vec}(\boldsymbol{R}_{k+1}))$$

$$\beta_{k+1}=\frac{\mathrm{tr}[(\boldsymbol{Z}_{k+1}^{\mathrm{T}}\boldsymbol{R}_{k+1})]}{\mathrm{tr}[(\boldsymbol{Z}_k^{\mathrm{T}}\boldsymbol{R}_k)]},$$

$$\boldsymbol{P}_{k+1}=\boldsymbol{R}_{k+1}+\beta_{k+1}\boldsymbol{P}_k,$$

$$k:=k+1.$$

算法2中选取不同的预处理矩阵可以得到不同的预处理算法。本文考虑对角预处理方法（DIAGCG）和SSOR预处理方法（SSORCG），即分别取

K=$\boldsymbol{D}_0$，

$$\boldsymbol{K}=\frac{1}{\omega(2-\omega)}(\boldsymbol{D}_0-\omega\boldsymbol{L})\boldsymbol{D}^{\mathrm{T}}(\boldsymbol{D}_0-\omega\boldsymbol{L}^{\mathrm{T}}),$$

其中$\boldsymbol{D}_0$为定理2中矩阵$\boldsymbol{M}$的对角元，$\boldsymbol{L}$为矩阵$\boldsymbol{M}$的严格下三角部分，ω为实常数。这2种预处理方法各有优点，对角预处理能较SSOR预处理节省计算时间，但并不能明显地降低条件数。而SSOR可以较明显地降低条件数，但不适合较大规模的数值问题。

4 数值试验

对本文提出的算法1和2种预处理算法，与现有的广义Schur算法（GSD）[8]和迭代方法算法（IMWCW）[4]在MATLAB中作比较。设初始迭代矩阵$\boldsymbol{X}_0$=0。当迭代超过4×10^4次或$\|\boldsymbol{R}_k\|<10^{-7}$时，终止算法。

第一个数值实例来自文献[9]，设问题1的系数矩阵为：

$\boldsymbol{A}$=diag$(1,\cdots,n)$+$\boldsymbol{U}_n$，

$\boldsymbol{B}$=$\boldsymbol{C}$=$\boldsymbol{I}_n$+0.5$\boldsymbol{U}_n$，

$\boldsymbol{D}$=0.5$\boldsymbol{I}_n$− diag$(n,\cdots,1)$+$\boldsymbol{U}_n$，

其中$\boldsymbol{U}_n$为元素为1的严格下三角矩阵。设$\boldsymbol{X}^*$为元素全为1的矩阵，选取$\boldsymbol{E}$为使$\boldsymbol{AX}^*\boldsymbol{B}+\boldsymbol{CX}^*\boldsymbol{D}=\boldsymbol{E}$成立的矩阵。图1给出了算法1（CG）和2种预处理算法（DIAG，SSOR）以及算法IMWCW[4]对问题维数与迭代次数的关系。

从图1中可以看出算法IMWCW较算法1的迭代次数多，而2种预处理方法，特别是SSOR预处理能够有效地降低原算法1的迭代次数。进一步对算法1和算法IMWCW作比较，表1列出了2种算法关于运算时间t和迭代次数n的比较。“--”代表迭代次数超过

40 000次。

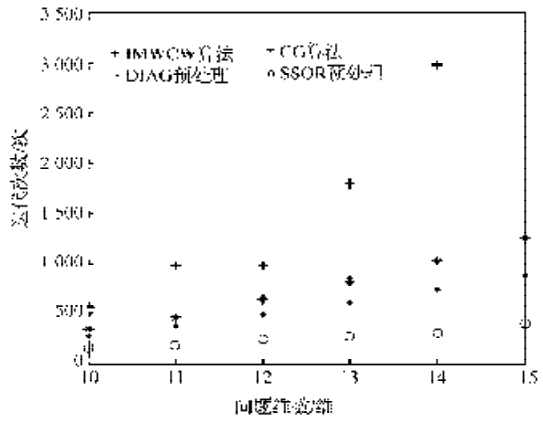

图 1　不同算法的迭代次数比较

Fig. 1　Comparison of the number of iterations between different Algs

表 1　算法 1 和算法 IMWCW 的比较

Table1　Comparison between Alg. 1 and Alg. IMWCW

问题维数	算法 1		算法 IMWCW	
/维	n/次	t/s	n/次	t/s
28	4 312	3.359	21 454	18.984
29	5 099	4.859	19 870	22.125
30	5 267	4.953	28 729	21.265
31	5 979	6.562	29 359	27.390
32	6 586	5.703	30 102	19.953
33	7 894	8.734	30 568	29.203
34	7 612	8.406	28 092	25.281
35	8 682	12.437	33 059	40.765
36	8 659	10.606	------	------
37	8 700	12.281	------	------

从表 1 中可以看出算法 1 要优于算法 IMWCW。

为了说明算法 1 对较大规模问题计算较广义 Schur 算法的优越性，考虑来自用 Newton 法解二次矩阵方程的子问题数值实例[8]。这时问题 1 的系数矩阵为

$A=D=I_n$，

$$B=\left(\frac{\|R\|_F+\sqrt{\|R\|_F^2+4\|T\|_F}}{2}\right)I_n,$$

$C=B+R$，$E=-B^2-RB-T$，

其中 R=tridiag(−10，30，−10)，T=tridiag(−5，15，−5)。图 2 给出了 2 种算法解问题 1 的维数和运算时间的关系。从中可以看出 CG 算法在解二次矩阵方程子问题时要优于广义 Schur 算法。

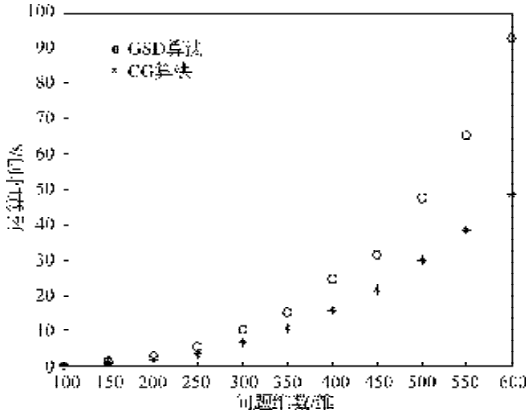

图 2　不同算法的运算时间比较

Fig. 2　Comparison on the CPU time between different Algs

参考文献：

[1] Chu K W. The Solution of the Matrix Equations $AXB+CXD=E$ and $(YA-DZ, YC-BZ)=(E, F)$[J]. Linear Algebra Appl.，1987，93：93−105.

[2] Peng Z Y. An Iteration Method for the Least Squares Symmetric Solution of the Linear Matrix Equation $AXB=C$ [J]. Appl. Math. Comput.，2005，170：711−723.

[3] Lei Y，Liao A P. A Minimal Residual Algorithm for the Inconsistent Matrix Equation $AXB=C$ over Symmetric Matrices[J]. Appl. Math. Comput.，2007，188：449−513.

[4] Wang M，Cheng X，Wei M. Iterative Algorithms for Solving the Matrix Equation $AXB+CX^TD=E$ [J]. Appl. Math. Comput.，2007，187：622−629.

[5] Golub G H，Van Loan C F. Matrix Computations[M]. 3rd ed. Baltimore：Johns Hopkins University Press，MD，1996.

[6] Saad Y. Iterative Methods for Sparse Linear Systems[M]. Boston：PWS，1996.

[7] 王松桂，吴密霞，贾忠贞. 矩阵不等式[M]. 北京：科学出版社，2006.
Wang Songgui，Wu Mixia，Jia Zhongzhen. Matrix Inequalities[M]. Beijing：Science Press，2006.

[8] Higham N J，Kim H M. Solving A Quadratic Matrix Equation by Newton's Method with Exact Line Search[J]. SIAM J. Matrix Anal. Appl.，2001，23：303−316.

[9] Gardiner J D，Laub A J，Amato J J，et al. Solution of the Sylvester Matrix Equation[J]. ACM Trans. Math. Software，1992，18：223−231.

（责任编辑：罗立宇）